Safe Realization of the Generalization Privacy Mechanism*

Tristan Allard, Benjamin Nguyen, Philippe Pucheral

PRiSM Laboratory,
Univ. of Versailles, France
⟨Fname.Lname⟩@prism.uvsq.fr

INRIA Rocquencourt,
Le Chesnay, France
⟨Fname.Lname⟩@inria.fr

***Abstract*—An increasing number of surveys and articles highlight the failure of database servers to keep confidential data *really* private. Even without considering their vulnerability against external or internal attacks, mere negligences often lead to privacy disasters. The advent of powerful smart portable tokens, combining the security of smart card microcontrollers with the storage capacity of NAND Flash chips, introduces today credible alternatives to the systematic centralization of personal data on servers. Individuals can now store their personal data (e.g., their medical folder) in their own smart tokens, kept under their control, and never disclose *in clear* their private data to the outside untrusted world. However, this new opportunity of managing and protecting personal data conflicts with the objective of implementing knowledge-based decision making tools on top of centralized data. This paper precisely addresses this issue and proposes to adapt the traditional *Generalization* privacy mechanism to an environment composed of a large set of tamper-resistant smart portable tokens seldom connected to a highly available but untrusted infrastructure. This combination of hypothesis makes the problem fundamentally different from any previously studied privacy-preserving data publishing problem we are aware of.**

I. Introduction

The Smart Token: an alternative to personal data centralization: Individuals are more and more reluctant to entrust their sensitive data to any data server. This suspicion is fueled by security surveys pointing out the vulnerability of database servers against external and internal attacks [13] and by many examples where negligence leads to personal data leaks (e.g., [4], [3]). This growing suspicion sometimes compromises nationwide projects: for instance, the Dutch Electronic Health Record program was canceled due to privacy concerns expressed by citizens [5].

In the meantime, credible alternatives to a systematic centralization of personal data on servers are arising. These alternatives build upon the emergence of new hardware devices called Secure Portable Tokens (SPTs for short). Whatever their form factor (SIM card, secure USB stick, wireless secure dongle), SPTs combine the tamper resistance of smart card microcontrollers with the storage capacity of NAND Flash chips. This unprecedented conjunction of portability, secure processing and Gigabytes-sized storage constitutes a real breakthrough in the secure management of personal data. Thanks to SPTs, personal records can be easily managed under the control of the record owner herself with security guarantees stronger than those provided by any central server. Today, the use of SPTs for e-governance (citizen card, driving license, passport, social security, transportation, education, etc) is actively investigated by many countries, and personal healthcare folders embedded in SPTs receive a growing interest, e.g., the Health eCard[1] in UK, the eGK card[2] in Germany, the LifeMed card[3] in the USA. As suggested in [6], SPTs can even embed highly versatile full-fledged personal data servers.

Reconciliation of Privacy with Knowledge-Based Decision Making: The counterpart of the privacy risks incurred by centralizing personal data on servers is the opportunity it offers for knowledge-based decision making and typically *privacy-preserving data publishing* (PPDP). A typical PPDP scenario starts by a **collection phase** where the *data publisher* (e.g., a hospital) collects data from *record owners* (e.g., patients), followed by a **construction phase** where the publisher computes the anonymization rules defining the transformations to apply to the collected data in order to anonymize it, and ends with an **anonymization phase** where the publisher effectively applies these rules to the data. The sanitized data thus produced can be released to a set of *data recipients* (e.g., a drug company, a public agency, or the public) for inquiry purposes. Most research in the PPDP area considers a model where the data publisher is trustworthy, so that record owners are assumed to easily consent providing it with their personal information [11]. As pointed out above, convincing record owners about the legitimacy of this trust assumption is difficult in practice.

Hence, governments and public agencies are faced today with two conflicting objectives: (1) the need for decision making tools, usually to increase a collective benefit (e.g., to prevent a pandemic thanks to an epidemiological study), (2) the obligation to get the consent of individuals to process their data electronically [1], pushing them to find alternatives to a systematic centralization of personal data (i.e., by using SPTs). In addition, the legislation in several countries authorizes statistical treatments of individuals' personal data without their explicit consent (assuming this consent has been given for the initial purpose of the data collection), provided that the data is adequately anonymized [1], [2]. While the spirit of the law is to protect the individuals' privacy better, the side effect is a new incentive for individuals to refuse their consent for the initial data collection if they distrust the way their data will be anonymized. Indeed, it does not make sense for an individual to consent to the management of her healthcare data to a SPT (because she distrusts central servers) while accepting that this same data will end up in a central server for anonymization purposes. The objective of this paper is precisely to address this

* This paper is an extended version of [9].

[1] http://www.healthecard.co.uk
[2] http://www.gematik.de
[3] http://www.lifemedid.com/

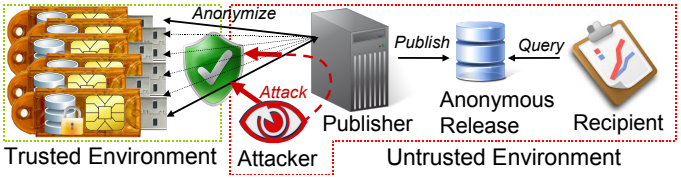


Fig. 1. Anonymous release of data stored on SPTs

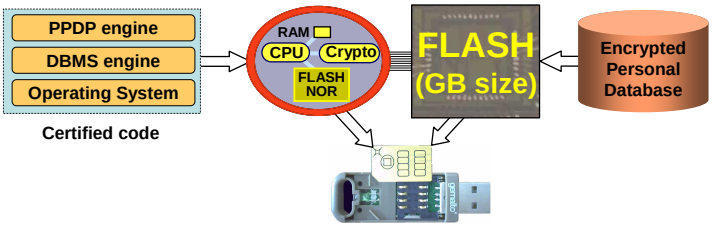


Fig. 2. SPT's internal architecture

issue, that is to safely (i.e., without privacy breaches) anonymize personal data hosted in SPTs while considering an untrusted PPDP model.

Motivating Scenario: Imagine a scenario where Alice carries her electronic healthcare folder on such an SPT. Alice's folder is then always available (accessible at any time, from any place and terminal, even in disconnected mode) and kept under Alice's control. Alices data never appears on any central server and no trace of interaction is ever stored in any terminal. If Alice loses her SPT, the SPT's tamper resistance renders potential attacks harmless. She can recover her folder from an encrypted archive stored by a trusted third party or managed by herself. If the health agency of Alice's country decides to collect sensitive data to perform an epidemiological study, Alice has no reason to be anxious because she has the assurance that (1) during the collection phase, her data will be exported only after having been encrypted by her SPT, and (2) during the anonymization phase, her data will be safely anonymized before being available in clear. Hence, no identifying data will be exposed with sensitive data on any central server. So, Alice can enjoy her healthcare folder with full confidence without compromising a collective healthcare benefit.

The above scenario is not futuristic. Medical-social folders embedded on SPTs are currently experimented in the Yvelines, a district of France, to provide care and social services at home to elderly people (PlugDB[4]). The folders mix medical and social data (income, dependent's allowance, marital status, entourage, food habits, etc) to the highest benefit of statistical studies. Being able to publish anonymized data from these folders is therefore a very important challenge. This challenge is however not restricted to the healthcare domain. Similar scenarios can be envisioned each time the legislation recognizes the right of the record owner to control under which conditions her personal data is stored and accessed.

Problem Positioning: Most PPDP research considers a trusted data publisher [11]. The untrusted case has been investigated in the context of Secure Multi-party Computation protocols (SMC) which allow several parties to jointly compute a function without revealing their input to one another. [21] presents a generic SMC approach, but unfortunately whose cost is exponential in the input size [12]. It is unusable for large datasets. Specific PPDP SMC protocols have been proposed [23], [24], [16]. However, they make strong assumptions on the attack model (e.g., introduction of a Trusted Third Party in [16], absence of collusion between the Publisher and a Helper Third Party in [23]) and on the communication model, requiring broadcasting messages among all parties. To the best of our knowledge, no previous work has ever considered the conjunction of hypothesis made in this paper: the tamper-resistance of the SPTs, their low availability and the untrustworthiness of the publisher.

Outline: The paper is organized as follows. Section II introduces the hypothesis our study relies on and states the problem. Sections III and IV discuss respectively the two attack models we must consider in our context, and propose data publishing protocols resistant to these attacks. Section V describes trivial alternatives to the proposed protocols and explains why they are inadequate. Section VI presents our experiments and demonstrates the practicability of the approach. Section VII intuitively explains why the techniques designed in this paper could be used to enforce privacy models other than k-anonymity. Finally, section VIII concludes by opening exciting research directions.

II. Problem Statement

This paper focuses on the organization of the collection and anonymization phases at the data source (i.e., at each SPT) while compromising neither privacy nor data utility compared to a trusted central server approach. The problem is difficult due to three assumptions: (1) the data publisher and the data recipients are untrusted, (2) the SPTs are trusted but there is no direct communication between them and (3) there is no certainty about the connection frequency and duration of each SPT connection.

Figure 1 illustrates the functional architecture and modus operandi. The *Trusted Environment (TE)* is constituted by the (possibly very large) set of SPTs. Each SPT hosts the personal data of a single record owner. It can take part in a distributed computation involving data issued from multiple record owners since all the SPTs trust each other. The *Untrusted Environment (UE)* encompasses the rest of the computing infrastructure, in particular the data publisher and the data recipients.

Hypothesis on TE: Regardless of their form factor (SIM card, secure USB stick, wireless secure dongle), SPTs are commonly made of a tamper-resistant microcontroller connected by a bus to a large external mass storage (Gigabytes of NAND Flash). A SPT can be seen as a basic but very cheap (today only a few dollars), highly portable, highly secure computer with reasonable storage and computing capacity for personal use. The trustworthiness of SPTs lies in the following factors:

- the SPT's embedded software inherits the tamper resistance of the microcontroller making hardware and side-channel attacks highly difficult,
- this software is certified according to the Common Criteria[5], making software attacks also highly difficult,

[4]http://www-smis.inria.fr/~DMSP/home.php

[5]http://www.commoncriteriaportal.org/

- this software can be made auto-administered thanks to its simplicity, contrary to its traditional multi-user server counterpart, thereby precluding DBA attacks,
- even the SPT owner cannot directly access the data stored locally; she must authenticate, thanks to a PIN code or a certificate, and only gets data according to her privileges.

For illustration purposes, Fig. 2 depicts the SPT used in the PlugDB project and in our experiments.

Hypothesis on UE: UE has unlimited computing power and storage capacity, and is available 24/7. The UE may have deviant behavior of two types:

- $Honest-but-Curious$: the attacker obeys the protocol it is participating in but tries to infer confidential data;
- $Weakly-Malicious$: the attacker has *weakly-malicious* intent [22] in that it cheats the protocol to disclose confidential data only if (1) the TE does not detect it and (2) the final result is correct.

Honest-but-Curious is an appropriate attack model for a well established data publisher (e.g., a government agency). The Weakly-Malicious model is better adapted to situations where the anonymization process is delegated to a third-party providing less guarantees. We do not consider Strongly Malicious attackers because the context of the study is such that the publisher is always liable of its actions and cannot take the risk of being detected. We assume in this paper that the SPT is not attacked. We show in [8] that, though highly improbable, hardware attacks can also be defeated by a mechanism guaranteeing a detection probability defeating *weakly-malicious* intent.

Hypothesis on the anonymization algorithm: We model the dataset to be anonymized as a single table $T(ID, QID, SD)$ where each tuple represents the information related to an individual hosted by a given SPT. ID is a set of attributes uniquely identifying an individual (e.g., a social security number). QID is a set of attributes, called *quasi-identifiers*, that could potentially identify an individual (e.g., a combination of Birthdate, Sex and Zipcode). The SD attributes contain sensitive data such as an illness in the case of medical records. The table schema, and more precisely the composition of QID and SD is application dependent.

In this article, for simplicity's sake, we consider the well studied *generalization* mechanism that drops ID and coarsens QID. This mechanism is used to enforce the k-anonymity privacy model [20], which consists in building *equivalence classes* of at least k tuples indistinguishable wrt their (generalized) QID. The tuples contained in an equivalence class are defined by the class's *generalization node*, that specifies an interval (numerical or categorical) for each dimension of the QID. Following the litterature's convention, we denote by $\succeq$ (resp. $\preceq$) the generalization (resp. specialization) relationship between a node and a QID (resp. a QID and a node). Figure 3 shows an example depicting raw data, their corresponding equivalence classes plotted in a 2D space, and the 2-anonymous dataset eventually delivered. Anonymizing the tuples whose QIDs are in the equivalence class EC_1 simply means replacing their QIDs by EC_1's generalization node, i.e., $([75001, 75002], [22, 31])$.

The approach proposed in this paper can work with any algorithm that keeps close semantics between an equivalence class and the values of the QIDs it contains. Most generalization-based algorithms fall into this category (e.g., [19], [15]). Extensions to other anonymization models is discussed in Section VII.

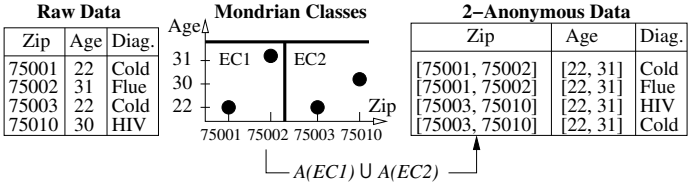


Fig. 3. 2-anonymous Equivalence Classes

Problem Statement: We address in this paper the problem of producing a k-anonymous version of a dataset defined by the union of the data hosted by a collection of SPTs (a subset of interest in TE) such that:

1) UE gets the final anonymized result but cannot learn anything else about individual's data;
2) The anonymized result is as useful (i.e., has the same quality) as if it had been computed by the same anonymization algorithm run by a traditional publisher on a central server.

The next sections tackle this problem for each of the *Honest-but-Curious* and *Weakly-Malicious* attack models.

III. Honest-but-Curious UE

Let us first consider the simplest attack model, namely *Honest-but-Curious*, where UE is assumed to fully respect the protocols defined but can make any inference or offline calculation in order to disclose the association between QIDs and SDs. We present below the proposed algorithm, called *Robust*, and defer to section V the discussion explaining the inadequacy of trivial solutions.

Robust's Phases: The Robust algorithm, shown in Alg.1, guarantees that the association $\langle QID_i, SD_i \rangle$, where i denotes an individual, remains hidden to UE while still allowing it to compute the equivalence classes. During the **collection phase**, the SPTs that connect send to UE tuples of the form $\langle QID_i, E_\kappa(SD_i) \rangle$, where E denotes a symmetric encryption scheme (e.g., based on the AES encryption function) taking a secret key κ as a parameter shared by all SPTs (key management is discussed next). When the UE decides that the collected sample of QIDs fits its needs (e.g., the size of the sample has reached a predefined threshold in the same way as traditional sampling in a centralized context), it stops the collection phase and launches the **construction phase**, during which it computes the equivalence classes based on the QIDs collected previously[6]. During the **anonymization phase**, *any SPT that connects* downloads a class (or more if its connection duration allows it), shuffles the class's tuples, and returns to UE anonymized tuples of the form $\langle E_\kappa^{-1}(E_\kappa(SD_i)) \rangle$. The shuffling step avoids UE to link a decrypted SD to its encrypted version based on its position in the returned result.

The Robust algorithm, summarized in Alg. 1, does not place any constraint on the availability of each SPT. First,

[6]Similarly to a central server context, the availability of data depends on the availability of their holders (e.g., the patients). Whether they are equipped with SPTs or not has no negative impact on data availability.

downloading and decrypting between k and $(2k-1)$ tuples does not present any bottleneck (in practice, k remains low - in the order of 10^2). Second, extending Robust to handle SPTs that get disconnected when they are treating a class (SPTs primarily serve other purposes than PPDP) is straightforward. We do not detail this extension here because it does not impact the core of the algorithm.

Algorithm 1 Robust Algorithm

Require: The k-anonymity level, the number s of QIDs required by the class construction phase, the encryption function E_κ parameterized by secret key κ shared among the SPTs.

1: **Collection phase:** For $i = 1,\ldots,s$, each SPT_i that connects sends its encrypted tuple $\langle QID_i, E_\kappa(SD_i)\rangle$ to UE.
2: **Construction phase:** UE computes EC, the set of equivalence classes respecting the k-anonymity privacy criterion.
3: Let $EC_j.T = \{\langle E_\kappa(SD_i)\rangle\}$ $s.t.$ $QID_i \preceq EC_j.\eta$ represent the set of tuples of the class $EC_j \in EC$.
4: **Anonymization phase:**
5: UE picks EC_j,a class not anonymized yet, and sends $EC_j.T$ to a connecting SPT_m.
6: SPT_m shuffles the tuples of $EC_j.T$.
7: **for all** $\langle E_\kappa(SD_i)\rangle \in EC_j.T$ **do**
8: SPT_m sends $\langle E_\kappa^{-1}(E_\kappa(SD))\rangle$ to UE
9: **end for**

Key management: The security of the Robust algorithm relies on the use of the secret key κ shared by all SPTs. We do the simplifying assumption that these keys are pre-installed by the SPT provider, though more dynamic protocols could be easily devised. Let us stress that even the SPT's owner cannot spy the hidden content and the computation made by her own SPT (in the same way as a banking card owner cannot gain access to the encryption keys pre-installed in his smart card microcontroller). Sharing secrets among all SPTs makes sense given the provable security guarantees they provide (see section II).

Correctness: k-anonymity is guaranteed by the fact that UE never gets access to a $\langle QID_i, SD_i\rangle$ tuple. During the collection phase, the only tuples it has at its disposal are in the form $\langle QID_i, E_\kappa(SD_i)\rangle$, with no way to decrypt SD_i. During the anonymization phase, UE learns the mapping between the set of tuples of each equivalence class, i.e., $\{\langle QID_i, E_\kappa(SD_i)\rangle\}$, and its corresponding set of returned sensitive data, i.e., $\{\langle SD_i\rangle\}$. Any QID in the former can correspond to any SD in the latter. Since each class contains at least k tuples, k-anonymity is safe.

IV. Weakly-Malicious UE

This section starts with an exhaustive list of the malicious actions upon which a *Weakly-Malicious* UE can base its attacks. The possible actions lie in tampering the data sent to the SPTs during the Anonymization phase in order to infer the links between QIDs and clear text decrypted SD values. Second, it upgrades the Robust algorithm with a set of *safety properties* preventing the weakly-malicious UE from acting maliciously.

A. Malicious Actions

Without loss of generality, any malicious action is either a *Destroy*, *Create*, or *Copy* action. The data upon which UE can apply these actions is the collected tuples.

Destroy and Create Actions: Destroying t tuple(s) in an equivalence class that contained k tuples leads to a $(k-t)$-anonymous class. For the same reason, creating t false tuples and adding them to $(k-t)$ collected tuples results in a class containing k tuples but that is in fact $(k-t)$-anonymous. In the following, we denote these actions **A1** and **A2** respectively.

Copy Actions: Tuples can be copied in two ways: either the UE produces a class that contains copies of the same set of tuples (intra-class copy, denoted **A3**), or it produces two classes, one containing a subset of tuples from the other (inter-class copy, denoted **A4**). Intra-class copies lead to a direct reduction of the k-anonymity level of the class, as previous actions do. Inter-class copies lead to inferences that are based on computing the differences between their respective SDs and QIDs. Indeed, (1) the SDs returned for both classes correspond to the collected QIDs belonging to both (the copied subset of tuples), and (2) the SDs returned for only one class correspond to the collected QIDs belonging to that class only. After having computed the differences, the UE is thus able to draw a correspondence between subsets of QIDs and SDs whose cardinality is less than k. These attacks are called *differential attacks*.

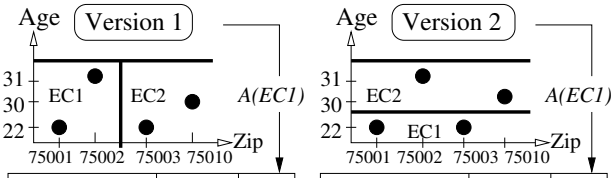
Fig. 4 depicts a differential attack. For instance, the version 2 of EC_1 contains one tuple copied from its first version and one new tuple. By computing the differences between the two versions, the attacker infers that (1) $QID = (75001, 22) \rightarrow SD = cold$, (2) $QID = (75002, 31) \rightarrow SD = flue$, and (3) $QID = (75003, 22) \rightarrow SD = HIV$.

B. Safety properties of equivalence classes

To prevent UE from acting maliciously, the equivalence classes must verify the following properties.

Local properties: Local properties are related to the content of each equivalence class, independently from the others:

- *Cardinality*: The given equivalence class contains at least k tuples.
- *Origin*: All tuples in the given equivalence class originate from a SPT.
- *Distinguishability*: All tuples in a given equivalence class are distinct.
- *Specialization*: The QID of each tuple specializes its class's generalization node. In other words, each tuple must



Zip	Age	Diag.
[75001, 75002]	[22, 31]	Cold
[75001, 75002]	[22, 31]	Flue

Zip	Age	Diag.
[75001, 75010]	[22, 22]	Cold
[75001, 75010]	[22, 22]	HIV

Fig. 4. An *overlapping* Differential Attack

belong to the proper class.

Global properties: Global properties stem from the relation of a given equivalence class with the whole set of classes already sent to an SPT:

- *Mutual Exclusion*: The Mutual Exclusion property requires that nodes of distinct classes generalize distinct sets of values.
- *Invariance*: The Invariance property requires that each class always be associated with the same content.

Coverage of Malicious Actions: The *Cardinality* property prevents the **A1** actions to endanger the k-anonymity of any equivalence class. The *Origin* property guarantees that any attempt of performing **A2** actions will be detected, and the *Distinguishability* property provides the same guarantee for **A3** actions. Together, the *Specialization*, *Mutual Exclusion*, and *Invariance* properties make **A4** actions inoperative. For space reasons, we do not detail the formal proof of the coverage, which is straightforward.

As a result, the safety properties defeat all malicious actions defined above, thereby precluding UE to launch any attack based on these actions. Processing equivalence classes satisfying all these properties will generate a k-anonymous result set with certainty.

C. Checking local properties

Checking the local properties in an SPT is rather straightforward. To test the *Cardinality* property, each SPT receiving an equivalence class during the anonymization phase checks that the number of tuples in the class is higher than k. To test the *Distinguishability* property, each SPT is assigned a unique identifier $SPTID_i$ which serves as tuple identifier TID_i. Tuple identifiers are encrypted with the tuples during the Collection phase, then each SPT participating in the Anonymization phase checks the uniqueness of this identifier in the equivalence class. To test the *Origin* property, a simple solution is for each SPT participating in the Collection phase to compute a MAC of its tuple. Finally, the MAC is checked by the SPTs participating in the Anonymization phase. To test the *Specialization* property, QIDs are encrypted within the tuples during the Collection phase, then each SPT participating in the Anonymization phase checks that the QIDs of all tuples specialize their class's node.

D. Checking global properties

No global history: Each SPT receives a single equivalence class per session, so checking the global properties *would* require that SPTs share information among them about the classes received. Unfortunately, SPTs are not able to communicate directly with each other: each SPT can solely rely on its own history. In the algorithm, UE can easily select the equivalence class sent to each SPT such that all the properties are satisfied from the SPT's viewpoint while they are violated from a global viewpoint. There is no ultimate solution to this problem since UE can delete any information sent by SPTs trying to build a common history or share a global viewpoint. However, considering that UE has weakly-malicious intentions, we propose to deter it from violating global properties by making any violation visible to SPTs through *caveat actions*.

Caveat actions: The first caveat action is to use anonymous communication channels between UE and SPTs [18]. This precludes UE to control which SPT receives which equivalence class. Consequently, the probability to send classes violating the global properties to the same SPT is no longer null. The second caveat action is to force UE to produce a *Summary* of the equivalence classes, which contains for every class its generalization node plus a digest of its content (e.g., a hash of its tuples). Each SPT participating in the anonymization phase primarily downloads the Summary $\mathcal{S}$, asserts the global properties based on $\mathcal{S}$, downloads a class, and checks the consistency between $\mathcal{S}$ and the downloaded class. If UE corrupts the content of a class, it will have to cascade the corruption to $\mathcal{S}$ in order to ensure its consistency with the class, and send the corrupted $\mathcal{S}$ to all the connecting SPTs. Any SPT receiving two disagreeing summaries will detect the attack. As a result, the detection is probabilistic, and its probability depends on the number of SPTs receiving each version of the summary. We have shown that the detection probability can be brought to highly deterring values (e.g., over 0.99) by tuning the minimal number of SPTs receiving each class. We refer the reader to [7] for a detailed discussion on this point.

Minimizing the cost of *Mutual Exclusion*: Although smart implementations of the *Mutual Exclusion* property can be designed in order to avoid a nested-loop style comparison of classes (e.g., in a sort-merge fashion), *Mutual Exclusion* remains one of the most costly checks. However, by slightly extending the *Invariance* property to encompass the classes's nodes in addition to their content, we can avoid the cost of checking *Mutual Exclusion* between summaries received at different moments. Indeed, if during its first connection, a SPT checks that the summary asserts *Mutual Exclusion*, it has only to check that the summary never changes during its following connections to guarantee that classes never overlap.

E. SPT algorithm for Weakly-Malicious UE

Algorithm 2 details the anonymization phase of the algorithm to be executed by each SPT. If a property check is not fulfilled, the SPT stops the execution and raises an alarm (e.g., to the destination of the SPT owner or a trusted third party). Due to lack of space, we do not detail the mechanisms (1) used to avoid an SPT from downloading an equivalence class already fully processed, and (2) used to assert that each class has been sent to enough SPTs to guarantee the desired detection probability.

V. Inadequacy of Trivial Solutions

Trivial alternatives to the Robust and WM algorithms could be devised based on collecting QIDs and SDs separately. Indeed, one could imagine to reorganize the phases as follows: (1) a phase of QID collection during which all SPT owners send their QID to the UE, followed by (2) a phase of construction during which the UE constructs the equivalence classes, and (3) a phase of SD collection during which each SPT that connects downloads the boundaries of equivalence classes and sends to UE its SD and the identifier of the class corresponding to its own QID.

Algorithm 2 Weakly-Malicious - SPT's Side

Require: An anonymous communication channel between the SPTs and UE, the k-anonymity level, E_{κ_1} and M_{κ_2} the encryption and MAC functions parametrized by secret keys κ_1 and κ_2 shared among the SPTs, a hash function H, and a function L returning the raw QID values that specialize a given generalizatin node.

1: Receive the current Summary $\mathcal{S}$: let $\mathcal{S}.EC$ denote the classes of $\mathcal{S}$, and $EC_i.\delta$ and $EC_i.\eta$ respectively the digest and generalization node of the class EC_i;
2: **if** $\nexists$ previous summary $\mathcal{S}_p$ **then**
3: **for all** $EC_i, EC_j \in \mathcal{S}.EC^2$ *s.t.* $EC_i.\eta \neq EC_j.\eta$ **do**
4: Check the *Mutual Exclusion* property: $L(EC_i.\eta) \cap L(EC_j.\eta) = \varnothing$;
5: **end for**
6: **else**
7: Check the *Invariance* of the number of classes: $|\mathcal{S}.EC| = |\mathcal{S}_p.EC|$;
8: **for all** $EC_i \in \mathcal{S}$ **do**
9: Check the *Invariance* of the classes's nodes and contents: $\exists EC_j \in \mathcal{S}_p$ *s.t.* $EC_i.\eta = EC_j.\eta$ and $EC_i.\delta = EC_j.\delta$;
10: **end for**
11: **end if**
12: Download a class EC_i;
13: Check the consistency between $\mathcal{S}$ and the class's content: $EC_i.\delta = H(EC_i.T)$;
14: Shuffle $EC_i.T$;
15: Check the *Cardinality* property: $|EC_i.T| \geq k$;
16: Init. the TIDs and decrypted tuples sets: $\Theta \leftarrow \varnothing$, $\Delta \leftarrow \varnothing$;
17: **for all** $t \in EC_i.T$ **do**
18: $d \leftarrow E^{-1}_{\kappa_1}(t)$;
19: Check the *Origin* property: $M_{\kappa_2}(d) = t.MAC$
20: Check the *Specialization* property: $d.QID \preceq EC_i.\eta$;
21: Check the *Distinguishability* property: $d.TID \notin \Theta$;
22: $\Theta \leftarrow \Theta \cup d.TID$;
23: $\Delta \leftarrow \Delta \cup d$;
24: **end for**
25: **for all** $d \in \Delta$ **do**
26: Send to UE $\langle d.SD \rangle$;
27: **end for**

Although the above scheme tackles the Honest-but-Curious attack model without requiring the sharing of any cryptographical material between SPTs, it still requires the enforcement of safety properties in order to cope with Weakly-Malicious attackers. Indeed, during the third phase, a Weakly-Malicious UE could send cheated classes spanning less than k individuals. The SPTs would thus still have to check the safety of the classes received during the third phase (e.g., distinguishability of tuples). Moreover, the above scheme incurs either an unbounded latency (if UE wishes to collect the SDs of *all* the SPTs having participated to the QID collection), or a discrepancy between the collected QIDs and SDs (otherwise). In order to avoid these drawbacks, the proposed Robust and WM algorithms collect both QIDs and SDs during the first phase.

VI. Experimental Validation

A. Experimental platform

The algorithms presented in this paper have been implemented and are being integrated in a larger prototype named PlugDB[4]. PlugDB aims at managing secure portable medical-social folders with the objective to increase quality and coordination of care provided at home to dependent patients. A complete chain of software (web server, application and DBMS server) has been developed and is embedded in the secure USB Flash platform pictured in Figure 2. This prototype has been demonstrated at [10]. The hardware platform is provided by Gemalto (the world leader in smart-cards), industrial partner of the project. The project is founded by the Yvelines District of France and by the French National Research Agency and will soon be experimented in the field in a medical network handling elderly people. The hardware platform is still under test so the performance measurements have been conducted on a cycle-accurate hardware emulator.

The algorithms considered for the experiments are *Robust* and *WM* (weakly-malicious). As a comparison baseline we also implemented the trivial algorithm described in section V and consisting in (1) collecting QIDs, (2) constructing the equivalence classes, and (3) collecting the SDs corresponding to all the QIDs collected first. We call this later algorithm *Naïve*.

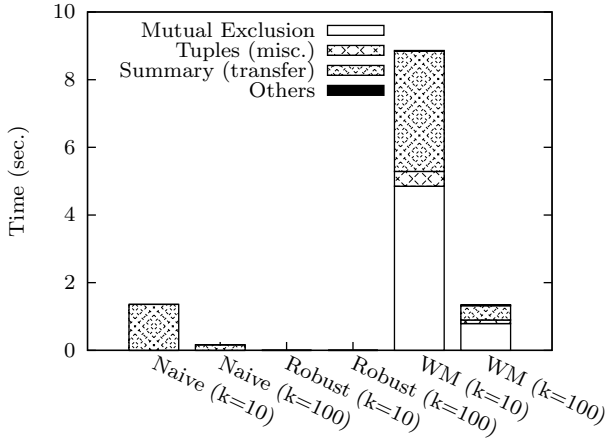
We concentrate on the evaluation (1) of the time spent internally in each SPT to participate to each phase of the protocol, and (2) of the protocol latency. We obtained the results of point (1) by performance measurements conducted on the hardware emulator, and the results of point (2) by simulation.

B. Internal Time Consumption

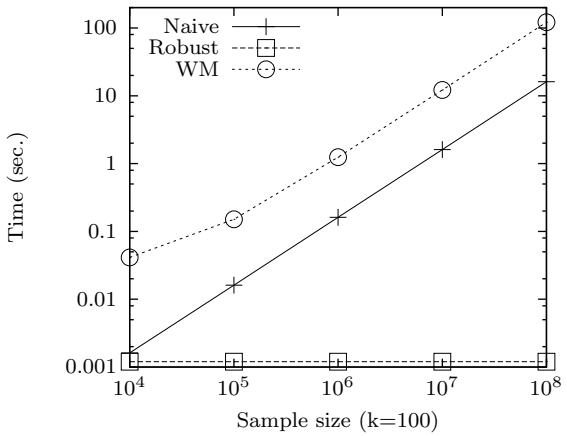
Settings: The cycle-accurate hardware simulator we used for this experiment is clocked at 50Mhz, corresponding to the CPU clock of the target platform. Cryptographic operations are implemented in hardware with good performances (e.g., encrypting a block of 128bits with AES costs 150 cycles). Although Hi-Speed USB2 (480 Mbps theoretical bandwidth) is announced for the near future, today's implementation of the communication channel is far less efficient. The measured throughput is 12Mbps (i.e., Full-Speed USB2), which amounts to 8Mbps of useful bandwidth when we exclude the overhead of the USB protocol itself.

Internal time consumption: Figure 5(a) details the time consumed by a SPT for each basic operation performed during the anonymization protocol. The measure has been performed with a sample of 10^6 SPTs, k varying from 10 to 100. The dataset was synthetically generated; two numerical attributes formed the QID and one string attribute formed the SD.

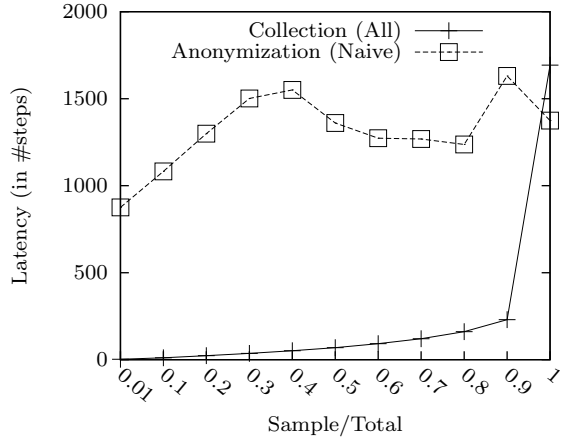
Depending on the algorithm, the worst case occurs either when k is minimal or maximal. For each algorithm, we plot these two cases to assess whether performance bottlenecks could compromise the feasibility of the approach. The worst case for Naïve and WM occurs when k is low. In this situation, the transfer cost of the Summary for both, and the checking cost of *Mutual Exclusion* for WM only, dominate the other costs because of the high number of equivalence classes. Note that



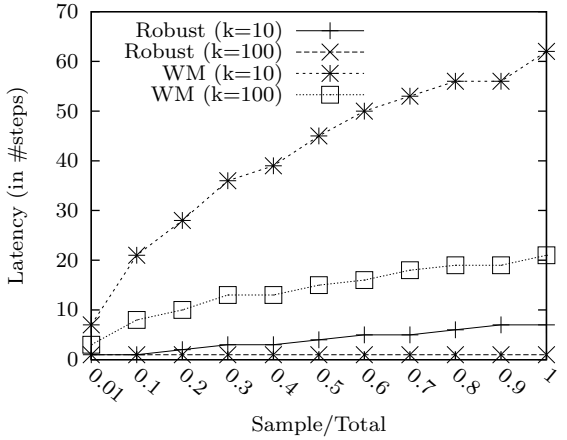
(a) Internal Time Consumption (details)



(b) Internal Time Consumption (scaling)



(c) Latencies: Collection Phase of all and Anonymization Phase of Naïve



(d) Latencies: Anonymization Phases of Robust and WM

Fig. 5. SPT's Time Consumption

a SPT only checks *Mutual Exclusion* once, i.e., at its first connection. It then checks *Invariance* during its subsequent connections (see section IV). Operations related to tuples (ie, transfer, hashing, and decryption) are cheap since Naïve solely uploads its sensitive data, and WM downloads and uploads between k and $2k-1$ tuples. On the contrary, the worst case for Robust occurs for a high k value, where the tuples' transfer cost overwhelms the other costs. Indeed, since Robust does not make use of any summary, its cost does not depend on the number of classes but on the cardinality of each class. As a conclusion, this figure confirms the feasibility of the approach by showing that, even in the worst cases, the execution time amounts to couples of seconds.

Scaling: Figure 5(b) shows the scaling of all the protocols wrt to the number of SPTs in the sample - chosen to be on a *nation-wide scale* - with $k = 100$. Apparently, Naïve and WM scale linearly with the number of SPTs sampled. This is due to the linear increase in size of the Summary (cost of transferring it and checking the global properties). Robust remains constant, around 10^{-3} sec.; indeed, it does not use any summary so the time it consumes only depends on k.

C. Latencies

Figures 5(c) and 5(d) plot respectively the latency of the Collection phase and of the Anonymization phase of the protocols, considering a population of 10^6 SPTs. The latency is measured in terms of connection steps of equal duration, this duration being application dependent. At each given step, each SPT SPT_i has a probability $\mathbb{P}_i$ of connecting to UE and executing the algorithm. We plot and compare below the latencies corresponding to a uniform connection distribution, where $\forall\ SPT_i, \mathbb{P}_i = 0.01$.

Collection phase (all) / Anonymization phase (Naïve): The latency of the collection phase is the same, regardless of the protocol studied. This latency depends on the connectivity distribution and on the proportion of SPTs in the sample. Figure 5(c) shows that the latency is about 160 steps when considering a sample of 80% of the total 10^6 SPTs. Note that this latency does not vary much with the total number of SPTs, and depends on $\mathbb{P}_i$ because the less often SPTs connect, the longer the protocol will be. On the same figure, since the times involved are of the same magnitude, we have also plotted the latency of the anonymization phase of the Naïve algorithm. This latency is about 1000 steps, regardless of the proportion

of SPTs reconnecting (and would be even bigger for a skewed distribution). These high numbers are explained by the fact that the same set of SPTs must connect at each phase of the protocol.

Anonymization phase (Robust and WM): Figure 5(d) shows the latency of the anonymization phase of the Robust and WM algorithms for $k = 10$ and $k = 100$. For Robust, we assumed that a connecting SPT anonymizes exactly one class during its session. The latency is linear and depends on the total number of classes to anonymize divided by the number of connected SPTs per step. The Robust's latency is constant and equal to 1 for $k = 100$ because there are more SPTs that connect during one step than the total number of classes. The WM's latency behaves also linearly. It differs from the Robust's one in that its increased protection incurs the supplementary cost of sending each class to several SPTs in order to guarantee the desired detection probability (in the measures, the minimal detection probability was set to 0.99).

As a conclusion, it appears from these figures that the latency of the Robust and Weakly-Malicious protocols is determined by the latency of their collection phase, itself being related to the size of the sample of interest in the complete population of SPTs. Note that we do not plot the latencies of a skewed distribution merely because it presents a limited interest ((1) the latency of the Collection phase depends on the number and connection probabilities of SPTs that connect few because UE may have to wait for their connection to reach the desired sample size, and (2) the latency of the Anonymization phase depends on the number of SPTs that connect at each step).

VII. Enabling other Privacy Models

The diversity of data recipients in terms of usage and trust preclude the election of a "one-size-fits-all" model. The enforcement of the k-anonymity model in a context where data is hosted on smart tokens is a first step towards the design of a broad framework able to adapt to various privacy criterion (e.g., l-diversity [17], PRAM [14]). For this, the key enabler lies in broadening the scope of the techniques explained in this paper, to privacy mechanisms other than generalization. This objective can be met in the following way. First, PPDP aims at obfuscating the link between identifying and sensitive data: some mechanisms act on the identifying part of data (e.g., generalizing the quasi-identifiers), some others on the sensitive part (e.g., perturbing the sensitive data). In our context, it appears that the construction phase, run by UE, can act indistinctly on either the former or the latter. Second, differential attacks are *not* specific to any mechanism. Their threat appears as soon as the collected dataset is divided into smaller partitions, which is necessary in a context where the sanitization task is distributed to light trusted devices that are unable to treat the dataset as a whole. Hence, (light variations of) the safety properties defined in this paper are still necessary for securing the realization of other mechanisms.

VIII. Concluding remarks

The increasing suspicion on the ability of DB servers to protect data against attacks and negligences urge the DB community to design credible alternatives to the centralization of personal data. This paper considers a new environment, where private data is stored by individuals into tamper-resistant smart portable tokens under their control. Unfortunately, this individual-centric environment conflicts with the collective requirement for knowledge-based decision making. This paper shows how to reconcile the best of the two worlds. To this end, we propose new secure distributed PPDP algorithms coping with the smart token's limited availability and the outside world's untrustworthiness.

This work paves the way for a new family of privacy-preserving distributed protocols exploiting the emergence of more and more powerful smart tokens. Future work will mainly consist in generalizing the approach to a wider variety of privacy mechanisms. The results presented in this paper are a strong incentive to go in this direction.

References

[1] European Parliament and Council: Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data, 1995.

[2] US Dept. of HHS: Standards for privacy of individually identifiable health information; final rule, 2002.

[3] Times Online. UK government loses personal data on 25 million citizens, November 2007.

[4] FierceHealthIT news. GA hospital health data breach due to outsourcing error, Sept. 2008.

[5] ICMCC. Dutch nationwide EHR postponed. Are they in good company?, 2009.

[6] T. Allard, N. Anciaux, L. Bouganim, Y. Guo, L. Le Folgoc, B. Nguyen, P. Pucheral, I. Ray, I. Ray, and S. Yin. Secure personal data servers: a vision paper. In *VLDB*, 2010.

[7] T. Allard, B. Nguyen, and P. Pucheral. Safe anonymization of data hosted in smart tokens. Technical Report 2010/25, PRISM Laboratory, 2010.

[8] T. Allard, B. Nguyen, and P. Pucheral. Sanitizing microdata without leak: Combining preventive and curative actions. In *ISPEC*, 2011.

[9] T. Allard, B. Nguyen, and P. Pucheral. Towards a safe realization of privacy-preserving data publishing mechanisms. In *MDM (PhD Colloquium)*, 2011.

[10] N. Anciaux, L. Bouganim, Y. Guo, P. Pucheral, J.-J. Vandewalle, and S. Yin. Pluggable personal data servers. In *SIGMOD*, 2010.

[11] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Comp. Surveys*, 2010.

[12] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the ACM Symp. on Theory of Computing*, 1987.

[13] L. A. Gordon, M. P. Loeb, W. Lucyshin, and R. Richardson. CSI/FBI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2006.

[14] J. Gouweleeuw, P. Kooiman, L. Willenborg, and P.-P. de Wolf. Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14, 1998.

[15] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, 2005.

[16] F. Li, J. Ma, and J.-h. Li. Distributed anonymous data perturbation method for privacy-preserving data mining. *J. of Zhejiang University*, 10(7), 2009.

[17] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[18] J. Ren and J. Wu. Survey on anonymous communications in computer networks. *Comput. Commun.*, 33:420–431, 2010.

[19] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6), 2001.

[20] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002.

[21] A. C. Yao. Protocols for secure computations. In *Proc. of the Symp. on Foundations of Computer Science*, 1982.

[22] N. Zhang and W. Zhao. Distributed privacy preserving information sharing. In *VLDB*, 2005.

[23] S. Zhong, Z. Yang, and T. Chen. k-anonymous data collection. *Inf. Sci.*, 179(17), 2009.

[24] S. Zhong, Z. Yang, and R. N. Wright. Privacy-enhancing k-anonymization of customer data. In *PODS*, 2005.